VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Univerzální administrační systém farnosti, část 1

Parish universal administration system, part 1

2010 **Jindřich Kuchař**

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Datum, podpis

Abstrakt

Práce se zabývá vývojem aplikace pro potřeby webové prezentace farností a děkanátu. Obsahuje popis jednotlivých prostředků pro tvorbu webových aplikací, spoluprácí skriptovacího jazyka PHP s databází MySQL (včetně testů rychlosti jednotlivých možností), rozebírá bezpečnost na internetu, nejčastější útoky proti webovým aplikacím a obranu proti nim. Dále popisuje import akcí z webu www.signaly.cz. Součástí této práce je také aplikace systému Microsoft Windows, určená pro správu intencí a ohlášek. Práce dále obsahuje téma věnující se programátorské dokumentaci.

Klíčová slova

PHP, ASP.NET, Python, MySQL, .NET, C#, vývoj, aplikace, internet, bezpečnost, XSS, CSRF, SQL Injection, iCalendar, programátorská dokumentace, OOP, třídy

Abstract

The work describes the development of a web application for the needs of a parish, which is intended to create a simple website. Contains a description of the means for creating web applications, collaboration of PHP language with a MySQL database (including speed tests), discusses security on the Internet, the most common attacks against Web applications and defending against them. It further describes events import from Web www.signaly.cz. Part of this work also discusses Windows application for work with notes and intention. The work also contains a theme dedicated to the programmer's documentation.

Key words

PHP, ASP.NET, Python, MySQL, .NET, C#, developing, application, internet, security, XSS, CSRF, SQL Injection, iCalendar, programmer's documentation, OOP, classes

Seznam použitých symbolů a zkratek

API	Application Programming Interface
ASP	Active Server Pages
CHM	Microsoft Compiled HTML Help
CLR	Common Language Runtime
DOM	Document Object Model
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IIS	Internet Information Services
LAMP	Linux, Apache HTTP Server, MySQL, PHP (kombinace technologií)
PDF	Portable Document Format
PDO	PHP Data Object
PHP	PHP: Hypertext Processor, dříve Personal Home Pages
SID	Session ID
SŘBD	Systém řízení báze dat
UASF	Univerzální Administrační Systém Farností
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

Obsah

1 Úvod

1.1 Cíl práce

Cílem práce je vývoj webové aplikace pro potřeby farností v České republice, především pak pro server www.farnost.cz, který poskytuje hosting a doménu pro stránky farností. Práce by měla navrhnout a implementovat jádro informačního systému, porovnat možnosti spolupráce PHP a MySQL, poukázat na možná bezpečnostní rizika internetových informačních systému a jejich řešení, popsat spolupráci s jinými webovými službami a také popsat možnosti programátorské dokumentace. V celém systému je třeba klást vysoký důraz na uživatelskou přívětivost. S touto práci úzce souvisí bakalářská práce studenta Michaela Hrabálka.

1.2 Vymezení pojmů

1.2.1 Internetový informační systém

Internetovým informačním systémem se rozumí systém umožňující práci s daty (sběr, zpracování, udržování a poskytování) v prostředí internetu. Provozovateli poskytuje jednoduché a pohodlné prostředí pro tvorbu nového obsahu a úpravu stávajícího. Návštěvníkům internetové prezentace pak poskytuje přehledné procházení tímto obsahem.

1.2.2 Dynamická webová aplikace

Dynamická webová aplikace umožňuje generovat internetové stránky na základě aktuálně dostupných informací. K tomu využívá skriptovací jazyky, a to jak na straně serveru, tak na straně klienta. Klientem je internetový prohlížeč, který v dnešní době často neslouží pouze jako pasivní zobrazovač, ale umožňuje vykonávat část logiky aplikace.

1.2.3 Systém řízení báze dat

Systém řízení báze dat (SŘBD) je softwarové vybavení, které zajišťuje efektivní práci s uloženými daty (databázi). SŘDB musí být navíc schopen řídit a definovat strukturu těchto dat. Současné systémy řízení báze dat navíc přidávají podporu pro autentizaci uživatelů, robustnost a zotavení při chybách, podporu jazyka vyšší úrovně jako je například SQL, transakce a mnoho dalšího. Mezi nejpoužívanější SŘBD v dnešní době patří Oracle, DB2, Microsoft SQL Server a PostgreSQL. Mezi tvůrci dynamických webových stránek je velmi populární MySQL v kombinaci s PHP. MySQL bude využitá při vývoji a provozu vyvíjené webové aplikace.

1.2.4 Administrátor systému

Administrátorem vyvíjené webové aplikace se rozumí osoba, která má možnost tvořit nový obsah a upravovat již vytvořený. Může se přitom jednat o osobu s velmi malou nebo žádnou zkušeností s tvorbou internetové prezentace, bez znalostí HTML.

1.2.5 Návštěvník webové aplikace

Návštěvníkem webové aplikace může být kterákoliv osoba s přístupem k internetu. Hlavním cílem návštěvy je snadné vyhledání informací, jako jsou kontakty na farní úřady, pořady bohoslužeb, pozvánky na pořádané akce a prohlídka fotografií z již uskutečněných akcí.

2 Prostředky pro tvorbu webové aplikace

Pro tvorbu dynamické webové aplikace je možné použit celou řadu technologií na straně serveru i klienta. Tato kapitola přináší základní přehled a popis technologií použitých při vývoji webové aplikace.

2.1 Serverové prostředky

Serverové prostředky, často ve formě skriptovacích jazyků, umožňují vytvořit obsah stránky tak, aby přesněji odpovídal požadavku uživatele. K uživateli je pak přenesen výsledek jejich práce, který se pouze zobrazí. Výhody tohoto řešení jsou

- Nízké nároky na software a hardware na straně klienta
- Nižší finanční náklady pro klienta
- Snadnější zabezpečení

Vzhledem k tomu, že je veškerá logika vykonávána na straně serveru, má toto řešení také několik omezení a nevýhod

- Vyšší nároky na výkon serveru
- Plná závislost na dostupnosti serveru
- Větší nároky na konektivitu

Mezi neznámější serverové technologie patří ASP.NET, Python a PHP.

2.1.1 ASP.NET

ASP.NET je součást .NET Frameworku společnosti Microsoft, určená pro vývoj a provoz internetových aplikací a služeb. Jeho historie se datuje zpět do roku 2001, kdy vznikl jako nástupce technologie ASP (Active Server Pages). ASP.NET je založen na CLR, díky čemuž umožňuje programátorům vyvíjet aplikace v jakémkoliv jazyku .NET (například C#, C++ a mnohé další). To umožňuje programátorům snadno přecházet mezi vývojem aplikací pro systém Windows a vývojem internetových aplikací. Díky tomu, že jsou aplikace ASP.NET automaticky předkompilované, mívají v porovnání se skriptovacími technologiemi vyšší výkon.

Výhody použití ASP.NET:

- Zjednodušený vývoj díky množství dostupných knihoven a ovládacích prvků
- Oddělení prezentační a logické části aplikace
- Automatické řešení bezstavosti protokolu http pomocí technologií ViewState a Session State (vyžaduje podporu na straně serveru)

Nevýhody použití ASP.NET:

- Přílišná provázanost s jinými technologiemi společnosti Microsoft
- ASP.NET 2.0 používá pro rozložení některých prvků na stránce tabulkové rozvržení
- Nekompatibilita session na ISS 6.0 a starších
- Omezená dostupnost hostování internetových aplikací zdarma (především v porovnání s PHP)

2.1.2 Python

Python je interpretovaný programovací jazyk, jehož původ sahá do roku 1989, kdy jej začal vyvíjet Guido van Rossum. Současná verze 3.0 pochází z roku 2008. Je podporován většinou běžných operačních systému (Microsoft Windows, Linux, MacOS). Jelikož se jedná o hybridní programovací jazyk, umožňuje používat jak objektové paradigma, tak procedurální a funkcionální. Jeho standardní knihovna obsahuje velké množství funkcí, díky čemuž je vývoj aplikací poměrně rychlý.

2.1.3 PHP

PHP: Hypertext Preprocessor (původně Personal Home Page) je skriptovací programovací jazyk určení pro programování dynamických webových aplikací. Je součástí technologie LAMP, kde spolu s operačním systémem Linux, webovým serverem Apache HTTP a databází MySQL tvoří open-source prostředí pro vývoj a provoz webových aplikací.

Počátky PHP sahají do roku 1995, kdy Rasmus Lerdorf začal pracovat na vývoji nástroje PHP/FI (Personal HomepageTools/Form Interpreter), což byla kolekce skriptů v jazyce Perl sloužící ke sledování přístupu na jeho stránky. Další funkcionalita byla již implementována v jazyce C a umožnila například komunikaci s databází. Výrazný vliv na pozdější rozšíření PHP bylo rozhodnutí o uvolnění zdrojového kódu, takže každý mohl přidávat novou funkcionalitu a opravovat existující chyby.

V roce 1997 vychází verze PHP/FI 2.0, která zaznamenala první výraznější rozšíření (instalace na přibližně 50 000[1] doménách). Přestože ke kódu přispělo několik dalších programátoru, stále se jednalo převážně o práci jednoho muže. Této verze si však všimli Andi Gutmans a Zeev Suraski, kteří hledali jazyk, který by jim usnadnil práci na jejich univerzitním projektu.

Jelikož PHP/FI 2.0 nesplňovalo plně jejich představy, kontaktovali Rasmuse a rozhodli se skriptovací jazyk zcela přepracovat. Vzniklo tak PHP 3, přímý předchůdce dnešního PHP. Velkou výhodou PHP 3 byla podpora rozšíření (modulů), díky kterým se k vývoji přidala velká skupina vývojářů. S touto verzí také vzniká kolem PHP komunita, která se stará nejen o jeho vývoj, ale také poskytuje zdarma množství již hotového kódu, vytváří dokumentaci a poskytuje návody pro tvorbu vlastních aplikací.

S PHP 4 přichází v roce 2000 nový pohled na zpracování kódu. Ten již není dále skriptovacím strojem překládán do strojového kódu, ale do bajtového kódu, jenž je pak zpracován pomocí stroje Zend Engine. Díky tomu se výkon jazyka výrazně zvýšil, přičemž zůstala zachována zpětná kompatibilita.

[1] PHP: History of PHP - Manual [online]. c2001, poslední revize 23. 4. 2010 [cit.2010-04-27]. Dostupné z: < http://www.php.net/manual/en/history.php.php>.

Současně používané PHP 5 přineslo novou verzi Zend Engine a vylepšený objektový model, podporu nových databázových funkcí a mnoho dalšího.

Výhody použití PHP:

- Jedná se o jazyk specializovaná pro vytváření webových aplikací
- Multiplatformnost
- Jednoduchá syntaxe
- Podpora na hostingových službách
- Svobodná licence

Nevýhody použití PHP:

- PHP jazyk není nikde definován
- Nekonzistentní pojmenování funkcí
- PHP sice podporuje výjimky, jeho knihovny je ale používají jen zřídka

PHP bude použito pro vyvíjenou webovou aplikaci.

2.1.4 Databáze MySQL

Při tvorbě dynamické webové aplikace je třeba řešit uskladnění dat a práci s nimi. Právě k tomu slouží nejrůznější SŘBD. V případě využití PHP se velmi často jedná o databázi MySQL.

MySQL je relační databázový systém vytvořený švédskou firmou AB, jehož původními autory jsou David Axmark, Allan Larsson a Michael Widenius. Počátky tohoto systému sahají do roku 1994, v současné době jej vlastní a vyvíjí firma Sun Microsystems vlastněná firmou Oracle. Jedná se o multiplatformní systém. Je součástí technologie LAMP, která poskytuje svobodný software pro vývoj dynamických webových aplikací. Mezi nejznámější webové služby využívající MySQL patří Facebook, YouTube a Wikipedia.

Spolupráci PHP a MySQL podrobněji rozebírá část MySQL.

2.2 Klientské prostředky

Vzhledem k rostoucím výkonům klientských počítačů, je možné přenést větší nebo menší část aplikační logiky blíže k uživateli. Klient pak již není pouze pasivním zobrazovatelem poskytnutých informací, ale je schopen sám reagovat na uživatelské vstupy. To umožní zjednodušit a zrychlit práci administrátora i uživatele. Mezi výhody klientských prostředků patří:

- Menší požadavky na server
- Možnost využití lokálních zdrojů
- Práce offline
- Nižší nároky na konektivitu

Nevýhody použití prostředků na straně klienta:

- Vyšší nároky na hardwarové a především softwarové prostředky klienta
- Různý stupeň podpory prostředku u klientů
- Nutnost přidat doplňky do klienta (prohlížeč) pro plnou podporu vyvíjené aplikace

Mezi nejpoužívanější klientské prostředky patří JavaScript, Adobe Flash, Java Applet a Microsoft Silverlight. Podrobnější informace o klientských prostředcích použitých při vývoji informačního systému lze získat ze související bakalářské práce vypracované studentem Michaelem Hrabálkem.

3 Bezpečnost internetových aplikací

V současné době zažívají internetové aplikace bouřlivý rozvoj. Díky moderním vývojovým nástrojům lze velmi snadno prezentovat svou práci širokému spektru uživatelů po celém světě. Mnohé internetové aplikace navíc umožňují zadávat a spravovat uživatelská data (ať již fotografie, blogy, nebo bankovní účty) odkudkoliv, kde je připojení k internetu. S rostoucí popularitou těchto aplikací roste také přitažlivost uchovávaných dat pro nejrůznější útočníky. Zatímco dříve měly útoky za cíl omezit či vyřadit z provozu aplikaci, případně znepříjemnit život uživateli, dnes je stále častějším cílem útoku získání citlivých dat určených k vydírání (např. soukromá korespondence, fotografie, zdravotní stav), nebo k ekonomickému prospěchu (čísla účtu, čísla bankovních karet, hesla atp.).

Bezpečnost internetových aplikací lze rozdělit do 4 části:

- Zabezpečení hardware
- Zabezpečení operačního systému a běžících aplikací
- Poučení uživatelé
- Zabezpečení samotné internetové aplikace

3.1 Zabezpečení hardware

Tato část zabezpečení spočívá v zamezení fyzického přístupu útočníka k serveru a diskům se zálohovanými daty. Servery jsou proto často umístěny ve speciálních místnostech, do kterých má přístup pouze nezbytně nutný počet lidí. Jednotlivé vstupy do místnosti je také vhodné monitorovat. Pro zvýšení bezpečnosti je třeba uložena data šifrovat, takže ani při případném přístupu k hardware nezíská útočník přímý přístup k datům.

3.2 Zabezpečení operačního systému a běžících aplikací

Nedílnou součásti zabezpečení internetové aplikace je zabezpečení operačního systému, na kterém běží vlastní internetová aplikace a zabezpečení ostatních aplikací běžících na tomto serveru. Do této části zabezpečení tak patří nejen pravidelná instalace záplat, ale také pečlivě oddělené uživatelské účty jednotlivých uživatelů s právy omezenými na nejnutnější minimum.

Pozornost je také třeba věnovat zabezpečení internetového serveru (např. Apache nebo IIS), se kterým komunikují uživatele (a útočníci).

3.3 Poučení uživatelé

Uživatelé představují největší bezpečnostní riziko nejen pro internetovou aplikaci, ale i sami pro sebe. Z pohledu tvůrců aplikace je tedy třeba omezit na minimum počet uživatelských vstupů do aplikace, důsledná kontrola vložených dat, množství uchovávaných dat o uživatelích (pro internetové album nemusí být například třeba uchovávat adresu jednotlivých uživatelů, případně jejich rodné číslo) a kdo má k uloženým datům přístup (například která data budou k dispozici pro indexaci vyhledávači). Lze

zde také zahrnout nároky na heslo (minimální délka, nutné použití písmen i číslic, doba vypršení hesla), u náročnějších aplikací zasílání potvrzovacích SMS atd.

Z pohledu uživatele se jedná především o rozumný přístup k poskytování osobních údajů (například hesla k jiným internetovým aplikacím, osobní fotografie, soukromý email), hlídání využití šifrovaného spojení (HTTPS) během zasílání citlivých údajů, atd.

3.4 Zabezpečení samotné internetové aplikace

3.4.1 Útoky proti serverové části

K útokům proti serverové části aplikace dochází při spuštění cizího kódu v kontextu aplikace. Tyto často velmi jednoduché útoky mohou mít dalekosáhle důsledky. Pokud se útočníkovi tento útok povede, získá stejná oprávnění, jako má samotná aplikace. Může tedy vytvářet a mazat soubory, vypsat zdrojový kód aplikace, přistupovat do databáze atd. Hlavním typem těchto útoků je PHP Injection a SQL Injection. Jelikož je při vývoji internetové aplikace využito PHP, budou tyto útoky popisovány právě na něm.

3.4.1.1 PHP Injection

Jedná se o zneužití funkcí `include()`, `require()` nebo `require_once()`, které na místo volání funkce vkládají obsah souboru v parametru. Pokud je obsahem souboru kód jazyka PHP, tento kód se vykoná.

Příklad špatného kódu:

```
http://www.mujweb.cz/index.php?zobraz=knihaNavstev.php
<?php include($_GET['zobraz']); ?>
```

V normálním případě je do souboru *index.php* vložen zdrojový kód obsahující logiku knihy návštěv. Útočníkovi však stačí pozměnit adresu zadanou do prohlížeče:

```
http://www.mujweb.cz/index.php?zobraz=http://hack.cz/mujKod.txt
```

Kde soubor *mujKod.txt* umístěný na libovolném serveru obsahuje libovolný kód jazyka PHP, který se má spustit na napadeném serveru. Často se jedná například o vypsání adresářové struktury nebo zdrojového kódu skriptu. Tato zranitelnost však nemusí být zneužitá pouze pro cílené omezení internetové aplikace, ale vložením skriptu rozesílajícího emaily může být využitá například pro rozesílání spamu.

Základem zabezpečení proti tomuto útoku přináší správné nastavení `magic_quotes_gpc()` (automatické escapování znaků), `allow_url_fopen()` (čtení dat přes HTTP), `allow_url_include()` (vkládání souborů přes HTTP) a `register_globals()` (automatické plnění proměnných). Toto nastavení útok značně stíží, ale nedokáže mu plně zabránit.

Účinným zabezpečením je kontrola proměnné obsahující načítaný skript pomocí `switch`:

```
switch ($_GET['zobraz'])
{
case "knihaNavstev.php":
include("knihaNavstev.php");
break;
case "kontakt.php":
include("kontakt.php");
break;
default:
include("hlavniStranka.php");
break;
}
```

Druhou možností, kterou může útočník využít pro provedení PHP Injection je špatně kontrolované nahrávání a stahování souboru. Pokud takto lze nahrát PHP skript a následně jej spustit, má útočník stejné možnosti, jako při zneužití funkcí `include()` a `require()`.

Obranou je v tomto případě možnost nahrávat pouze soubory určeného typu (podle přípony, například pouze .pdf nebo .doc), a dále zakázat v souboru *.htaccess* skriptování ve složce obsahující tyto nahrané soubory.

Na druhé straně je třeba dávat pozor na skript umožňující stahování souborů. Pokud by umožňoval stažení libovolného souboru, může si útočník pomocí využití adresářů . a .. stáhnout všechny skripty a zjistit tak například hesla pro připojení k databázi a podobně. Je tedy třeba opět omezit typy stahovaných souborů, případně oddělit „název" souboru v požadavku na stažení od skutečného názvu a cesty k němu (například pomocí využití databáze).

3.4.1.2 SQL Injection

Jedná se o techniku napadení databázové vrstvy aplikace vsunutím kódu přes neošetřený vstup a vykonání podvrženého dotazu v jazyku SQL na databázový server. Tento typ útoku se přitom neomezuje pouze na internetové aplikace, ale na všechny aplikace komunikující s databází a využívající jazyk SQL. Stačí přitom mít možnost editovat pouze část dotazu.

Příklad špatného kódu:

```
http://www.mujweb.cz/index.php?user=petr
select * from users where name= '$_POST[”user”]';
```

V normálním případě vrátí databázový server řádek tabulky se zvoleným uživatelem. Útočník si však může změnou adresy vypsat všechny uživatele (a jejich hesla):

```
http://www.mujweb.cz/index.php?user=petr' OR 1=1; --;
```

Skutečně vykonaný dotaz pak bude vypadat takto:

```
select * from users where name= 'petr' OR 1=1; --';
```

Takto je možné obejít například přihlášení uživatele nebo smazat celou tabulku. Díky využití klauzulí UNION a JOIN není útočník omezen pouze na jednu tabulku.

Mezi základní obrany proti tomuto útoku patří omezení dynamicky sestavovaných dotazů, kontrola očekávaných typů například pomocí funkcí `is_int()`, `is_string()` atd. Důležité je také ošetřování vstupních dat pomocí funkce `mysql_real_escape_string()` (převede vstupní řetězec do podoby bezpečné pro použití v SQL dotazu, v novějších verzích PHP však nebude tato funkce dostupná).

Skutečnou funkční ochranou je použití předpřipravených dotazů a následné vázání proměnných. V tomto případě je databázi nejprve předán dotaz s odkazy na parametry, a následně jsou databázi předané samotné parametry. Databáze pak ví, že tyto parametry nemá interpretovat a útok SQL Injection je tak vyloučen.

Při využití kombinace PHP a MySQL umožňují tento způsob vykonávání SQL dotazu třídy MySQLi a MySQLi_STMT:

```
$mysql = new mysqli();
…
$stmt = $mysqli->prepare(”select * from users where name= ?”);
/* 's' značí, že parametr je typu string */
$stmt->bind_param('s', $_POST['user']);
$stmt->exekute();
…
```

Výhodou tohoto řešení je také možnost opětovného spuštění uloženého dotazu se změněnými parametry, což může přinést nárůst výkonu, případně omezená kontrola typu vkládaného parametru. Nadále má však smysl kontrolovat typ vstupních dat pomocí výše popsaných funkcí.

Důležité je také kontrolování uložených hodnot, například na výskyt HTML kódu, který by mohl změnit výstup aplikace (například vložením JavaScriptové funkce). Tyto uložené hodnoty také mohou obsahovat kód pro vykonání SQL Injection v jiném, neošetřeném dotazu, jehož vstupní parametry jsou brány z databáze.

Nedílnou součásti je také správné nastavení přístupových práv, jako například zákaz mazáni tabulek, úpravy databázového schématu, zápis do konkrétních tabulek a podobně. Útočník tak má pouze omezené možnosti zneužití databáze.

3.4.2 Útoky proti uživateli internetové aplikace

Tyto útoky mají za cíl získat oprávnění uživatelů pracující s aplikací, ať už zjištěním hesla, nebo donucením uživatele vykonat akci, kterou by za normálních okolností vědomě nevykonal. Nic netušící uživatel si tak může vymazat svá (a pokud k tomu má oprávnění) i cizí data, sdělit útočníkovi přihlašovací údaje a podobně.

3.4.2.1 XSS – Cross-Site Scripting

Při tomto útoku se útočník snaží zaslat uživateli vlastní javacriptový kód. Proto se tato chyba vyskytuje velmi často tam, kde uživatele mohou zadávat data, která se dříve nebo později objeví na výstupu aplikace. Pokud se útočníkovi podaří spustit vlastní javascriptový kód, může se zobrazenou stránkou díky využití DOM dělat prakticky cokoliv. Často tak dochází k přesměrování uživatele na stránku útočníka, zasílání dat z formulářů (přihlašovací údaje) na útočníkův server a podobně. Pokud

navíc dojde k odcizení SID, může útočník vystupovat pod identitou uživatele. Přítomnost škodlivého kódu na stránce navíc nemusí být patrná na první pohled.

Příklad chybného kódu:

```
http://www.mujserver.cz/index.php?den=utery
<?php echo 'Zobrazený den: '.$_GET['den'];?>
```

Útočníkovi pak stačí nahradit utery za `<script>/*Nějaký kód JavaScript*/</script>` a může se stránkou provádět vše, co mu javascript dovolí.

XSS útok se dělí na tři základní skupiny:

- Persistent
- Non-persistent
- DOM-based XSS

Persistent se vyznačuje tím, že po úspěšném útoku zůstává javascriptový kód součástí stránky a provede se všem návštěvníkům, kteří tuto stránku navštíví. Lze jej provést například vložením příspěvku obsahující javascriptový kód do návštěvní knihy.

Non-persistent naproti tomu vkládá javascriptový kód do HTTP požadavku, často tedy vyžaduje spolupráci uživatele (například kliknutí na odkaz).

DOM-based XSS útok využívá pro vypsání škodlivého kódu vlastnosti DOM prohlížeče, který umožňuje pomocí javascriptu vypsat například hodnoty proměnných předaných pomocí požadavku GET.

Obranou je důsledné využití PHP funkce `htmlspecialchars()` nejen na vstupu od uživatele, ale především na výstupu. Při nutnosti použít HTML je vhodné povolit pouze konkrétní tagy a pečlivě kontrolovat jejich parametry, zda neobsahují javascriptový kód.

Klient se může proti tomuto útoku účinně bránit zákazem JavaScriptu v prohlížeči.

3.4.2.2 CSRF – Cross-Site Reques Foregy

Tento útok má za cíl provést nějakou akci v internetové aplikaci pod identitou oběti útoku. Velmi snadno může být veden proti aplikacím, ve kterých se informace o přihlášeném uživateli uchovávají v cookies. Nejčastěji útočník donutí uživatele použit speciální odkaz na skript internetové aplikace provádějící akci, kterou by uživatel sám neprovedl. Tento útok je často zneužívám spolu s XSS.

Příklad útoku CSRF (změna hesla, funkce `zmenHeslo()` řeší přihlášení uživatele pomocí cookies):

```
http://www.server.cz/zmenHeslo.php?heslo=123456
<?php zmenHeslo($_GET['noveHeslo'])?>
```

Útočníkovi pak stačí přimět přihlášeného uživatele k navštívení stránky obsahující obrázek (na příklad zasláním soukromého vzkazu):

```
<img src="http://www.server.cz/zmenHeslo.php?heslo=utok"  height=0"
width="0" />
```

Ochranou proti tomuto útoku je použití tzv. hlídacích lístků. Ke každému formuláři je přidáno skryté pole obsahující jednoznačný identifikátor umožňující provedení určité akce právě jednou.

```
<form action=”zmenHeslo.php method=”get”>
      <input type=”pass” name=”heslo” />
      <input type=”hidden” name=”token” value=”gIkgoeHdkgo78a5e6g” />
      <input type=”submit” value=”Změnit heslo” />
</form>
```

Identifikátor token, který je uložen v session nebo v databázi, je při každém načtení formuláře znovu generován. Skript provádějící aktualizaci hesla pak vypadá takto:

```
<?php if($_GET['token'] == $_SESSION['token']){
zmenHeslo($_GET['noveHeslo')]);
}
unset($_SESSION['token']); ?>
```

4 MySQL

4.1 Proč MySQL

Jak již bylo v úvodu řečeno, systém je vyvíjen především pro potřeby hostingu www.farnost.cz. Počítá se však také s jeho provozováním u jiných, především bezplatných poskytovatelů hostingu, u kterých tvoří kombinace PHP a MySQL výraznou většinu. Z tohoto důvodu nemá příliš smysl uvažovat o jiném databázovém systému.

4.2 MySQL a PHP

Kombinace PHP a MySQL je pro vývoj internetových prezentací velmi častá. PHP podporuje procedurální i objektový přístup k MySQL. Postupným vývojem vznikly pro PHP tři API, které umožňují práci s databázovým serverem MySQL. Jedná se o:

- Rozšíření MySQL
- Rozšíření mysqli
- PDO (PHP Data Object)

4.2.1 Rozšíření MySQL

Jedná se o nejstarší API poskytující přístup k MySQL pro webové aplikace vytvořené pomocí PHP. Nabízí pouze procedurální přístup a vzhledem ke svému stáří je určeno především pro MySQL verze starší než 4.1.3, v novějších verzích nepodporuje některé nové funkce jako například předpřipravené dotazy či kurzory. Výhodou je široký zástup vývojářů, kteří umí s tímto rozhraním pracovat, lety ověřené postupy a opravené chyby.

4.2.2 Rozšíření mysqli

Toto rozhraní, dostupné pouze pro PHP 5, je určeno pro použití s MySQL 4.1.1 a novějšími. Přináší podporu pro vázání proměnných, přípravu a spouštění dotazů, podporu kurzorů, podporu užití více příkazu v jednom dotazu a další. Podporuje jak procedurální, tak objektový přístup. Toto rozšíření je v současné době doporučováno[2].

[2] PHP: Overview - Manual [online]. c2001, poslední revize 23. 4. 2010 [cit.2010-04-27]. Dostupné z: < http://www.php.net/manual/en/mysqli.overview.php>.

4.2.3 PDO

PHP Data Object je abstraktní databázová vrstva pro PHP aplikace. Poskytuje konzistentní API pro PHP aplikace bez závislosti na použité databázi. Díky tomu teoreticky umožňuje změnu použité databáze bez výraznějších změn v samotné aplikaci. Nevýhodou je omezená podpora nejnovějších funkcí databázového serveru MySQL a poměrně velké množství chyb ve starších verzích PHP 5.

4.2.4 Třídy MySQL a MyStatementSql

Pří výběru používaného API budeme rozhodovat pouze mezi rozšířením MySQL, mysqli a mysqliE (třída dědící z mysqli). Jelikož je celý systém vyvíjen objektově, je nejprve třeba přepsat rozšíření MySQL na objekty, do kterých je přidáno několik rozšíření, jako například simulace předpřipravených dotazů, podpora výjimek nebo obrana proti útoku SQL injection. Výsledkem přepisu jsou dvě třídy:

- MySql, která zajišťuje připojení k databázovému serveru a výběr databáze. Navíc obsahuje dvě metody. `MySql->query()` a `MySql->prepare()`. Obě vrací objekt třídy MyStatementSql.
- Třída MyStatementSql, která vykonává samotný dotaz, přidává parametry zabezpečené proti SQL injection a umožňuje procházet výsledkem.

Třída MySql je velmi jednoduchá, v konstruktoru očekává parametry spojení (host, uživatel, heslo a databáze). Samotné připojení obstarává metoda `MySql->connect()`. Tuto metodu je možno volat ručně, nebo je automaticky zavolána při prvním dotazu na databázi. V případě že se připojení nezdaří, vyvolá tato metoda výjimku.

Metoda `MySql->query()` očekává jako jediný parametr sql dotaz, který pomocí dalších metod okamžitě provede. V případě chyby vyvolá výjimku, jinak vrací hodnotu *TRUE* pokud se jedná o dotaz typu *INSERT, UPDATE, DELETE* atd., nebo objekt třídy MyStatementSql, pokud se jedná o dotazy typu *SELECT, SHOW, DESCRIBE* atd. Tato metoda nijak neupravuje sql dotaz, proto je třeba při jejím používání klást zvláštní důraz na bezpečnost (nebezpečí SQL Injection).

Metoda `MySql->prepare()` očekává jako jediný parametr SQL dotaz, který připraví pro pozdější spuštění a přidání případných parametrů.

Podstatná část logiky vykonávaní dotazu se nachází ve třídě MyStatementSql, která v konstruktoru očekává databázové spojení a předpřipravený dotaz. V tomto dotazu lze pro upřesnění místa vloženi vázaných parametrů použít zápis *„:0”*, kde číslo za dvojtečkou značí pořadí parametru.

Vložení parametrů a provedení dotazu je provedeno zavoláním metody `MyStatementSql->execute()`. Tato metoda má proměnlivý počet parametrů a před svázáním těchto parametru s předpřipraveným dotazem na ně zavolá funkci `mysql_escape_string()`, čímž dojde k zamezení útoku SQL Injection.

4.2.5 Třídy MySQLiE a StatementSql

Třída MySQLiE dědí ze třídy MySQLi a přidává podporu výjimek. Navíc obsahuje metodu `MySQLiE->rowExist()`, která vrací hodnotu *TRUE* v případě, že řádek vybraný dotazem předaným v parametru této funkce existuje, v opačném případě vrací hodnotu *FALSE*. Tato funkce sama o sobě nenabízí ochranu proti SQL Injection, proto je třeba při jejím užitím dbát na zvýšenou opatrnost. S touto třídou lze díky dědičnosti pracovat jako se třídou MySQLi, díky čemuž je snadněji přístupná většímu množství programátoru. Metoda `MySQLiE->prepare()` vrací objekt třídy StatementSQL.

Třída StatementSql dědí ze třídy MySQLi_STMT, ke které přidává podporu výjimek a snadnější vázaní parametrů. Parametry pro svázaní s předpřipraveným dotazem lze vkládat přímo do metody `StatementSql->execute()`, navíc není třeba uvádět jejich typ a nemusí být předávaný jako ukazatel. Metoda `StatementSql->bindResult()` očekává jako jediný parametr pole ukazatelů na proměnné, do kterých se má uložit výsledek SQL dotazu.

4.2.6 Porovnání výkonu vlastních tříd a třídy mysqli

Vzhledem k tomu, že všechny výše popsané třídy přinášejí velmi podobnou funkčnost, je pro vybrání konkrétního používaného řešení třeba provést srovnávací testy. Při vyhodnocování testů bude kromě náročnosti použití výsledného řešení kladen důraz také na rychlost vykonávání dotazů.

Testování bude probíhat na tabulce users, která má stejnou strukturu jako tabulka použitá pro vyvíjený informační systém. Pro potřeby testování byla tato tabulka naplněna 1 000 000 náhodně generovaných záznamů. Testované dotazy vychází z těch, které používá třída UserUsr (viz část Základy systému). Každý dotaz byl na testovacím počítači spuštěn 10 000 krát a měření se desetkrát opakovalo. Všechny uváděné časy jsou v sekundách.

4.2.7 Naměřené hodnoty

Prvním testovaným dotazem je dotaz sloužící pro přihlášení uživatele:

```
SELECT `id_user`, `name` FROM `users` WHERE (`login`= ? AND `pass` =
sha1(?)) LIMIT 1;
```

Číslo měření	Třídy MySQL a MyStatementSql	Třídy MySQLi a MySQLi_STMT	Třídy MySQLiE a StatementSql
1	7,55	6,25	7,51
2	7,64	6,29	7,31
3	7,95	6,95	7,22
4	7,32	7,41	7,08

5	7,23	7,14	6,54
6	7,06	6,25	7,01
7	7,07	6,32	6,58
8	7,19	6,21	7,11
9	7,11	6,32	7,05
10	7,23	6,28	7,12
Průměr	**7,34**	**6,54**	**7,05**

Druhý testovaný dotaz vypadá takto:

```
UPDATE `users` SET email = ? WHERE id_user = ?;
```

Číslo měření	**Třídy MySQL a MyStatementSql**	**Třídy MySQLi a MySQLi_STMT**	**Třídy MySQLiE a StatementSql**
1	7,01	6,17	6,74
2	7,57	6,74	6,53
3	7,11	6,00	6,28
4	7,25	7,09	6,12
5	7,51	6,14	6,95
6	6,49	6,64	6,81
7	7,15	6,46	6,72
8	7,20	6,37	7,06
9	7,19	6,13	6,14
10	7,36	6,11	6,95
Průměr	**7,18**	**6,38**	**6,63**

Třetí testovaný dotaz slouží pro registraci nového uživatele:

```
INSERT INTO `users` (login, pass, name , email, reg_date) VALUES (?,
sha1(?), ?, ?, NOW());
```

Číslo měření	Třídy MySQL a MyStatementSql	Třídy MySQLi a MySQLi_STMT	Třídy MySQLiE a StatementSql
1	8,20	7,75	7,86
2	8,63	7,57	8,86
3	7,63	9,07	8,74
4	8,66	8,92	9,28
5	9,60	8,88	9,25
6	9,67	8,57	7,95
7	8,57	7,68	9,15
8	8,78	7,66	7,94
9	8,31	8,24	8,06
10	8,67	7,91	8,98
Průměr	**8,67**	**8,23**	**8,61**

Z naměřených hodnot lze vyčíst velmi vyrovnaný výkon jednotlivých možnosti propojení PHP a MySQL. Překvapením je velmi malý vliv předpřipravených dotazů na výkon. Ty tak přinášejí pouze větší bezpečnost. Jelikož pro tento administrační systém není očekáván přístup velkého množství uživatelů současně (ani obecně vysoké vytížení databáze), je možné vybírat ze všech tří testovaných řešení. Z důvodu budoucího vývoje a omezené podpory nových možností databáze MySQL nebude použito tříd MySQL a MyStatementSql. Pro jednodušší použití, podporu výjimek a pouze mírně nižší výkon bude při implementaci UASF použito tříd MySQLiE a StatementSql (pro potřeby vyvíjeného systému přejmenovány na MySQL a StatementSql).

4.3 Shrnutí databáze MySQL

Kombinace databáze MySQL a PHP je jedním z nejčastějších řešení při vývoji a provozu malých a středních internetových aplikací. Vyznačuje se především nízkou cenou, vysokým výkonem a rozsáhlou uživatelskou základnou. Nevýhodou je omezená, nebo téměř žádná podpora uložených procedur, transakcí atd.

V případě spolupráce s PHP lze volit ze 3 rozšíření, přičemž za nejvýhodnější a nejstabilnější je v současné době považována třída MySQLi a MySQLi_STMT. Ty přinášejí podporu nových funkcí

databáze MySQL jako jsou předpřipravené dotazy a vázání proměnných (čímž zamezují útoku SQL Injection).

Z testů vyplynulo, že z hlediska výkonu je mezi jednotlivými možnostmi spolupráce PHP a MySQL minimální rozdíl, a záleží tak tedy především na preferenci uživatelů.

5 Programátorská dokumentace

Ve zdrojovém kódu se mimo vlastní logiku aplikace mohou nacházet také dodatečné informace sloužící lidem, kteří daný kód čtou. Jedná se o tzv. komentáře. Tyto komentáře přitom nemusí sloužit pouze jiným osobám, ale mohou se hodit i samotným programátorům při návratu k dříve napsanému kódu.

Komentáře (nejen v PHP) mají přesně daný styl, díky kterému jdou snadno dohledat, mohou být barevně odlišeny vývojovým prostředím a hlavně kompilátor (respektive skriptovací stroj) ví, že tyto částí zdrojového souboru má vynechat z kompilace.

Ve většině programovacích jazyků je možnost použít dva druhy komentářů. Jednořádkový začínající symbolem dvou lomítek a končí s koncem řádků. Tento typ komentářů je vhodný pro okomentování konkrétního řádku kódu, u kterého nemusí být zcela jasná jeho funkce:

```
//Toto je jednořádkový komentář
```

Druhým typem je komentář víceřádkový, který se hodí pro popis většího bloku kódu, jako jsou například třídy, metody, funkce a podobně:

```
/* Toto je víceřádkový komentář,
*  který pokračuje na druhém
*  a třetím řádku
*/
```

Dohledávat tyto komentáře v jednotlivých souborech (kterých může být u velkých projektů několik stovek) však může být časově velmi náročné. Navíc není vždy potřeba znát přesnou strukturu funkce, její logiku a umístění souboru se zdrojovým kódem. Mnohem častěji si programátor vystačí se vstupními parametry a návratovou hodnotou (tzv. API). Z toho důvodu vznikly nástroje, které umí díky speciálním vloženým komentářům automaticky vygenerovat potřebnou dokumentaci, často ve formátu XML, HTML, PDF a podobně.

Při tvorbě internetové aplikace byly použity jazyky PHP, Java a C#. Z toho důvodu jsou možnosti automaticky generované dokumentace představeny právě na těchto jazycích.

5.1 PHPDoc

PHPDoc je typ komentářů kódu jazyka PHP, který vychází z Javadoc. Na rozdíl od použití obecných a náhodných komentářů, přináší tři hlavní výhody:

- Sjednocuje vzhled komentářů mezi programátory
- Umožňuje generátorům dokumentace (jako je phpDocumetor) vytvářet přehlednou a snadno pochopitelnou dokumentaci k API
- Umožňuje vývojovým prostředím (jako je NetBeans) lépe interpretovat typy proměnných pro slabě typované PHP

Samotné komentáře PHPDoc vycházejí z obecných víceřádkových komentářů, a nikterak tedy nenarušují zpracování PHP kódu skriptovacím strojem. Každý PHPDoc komentář začíná `/**` na

prvním řádku, každý další řádek začíná * a poslední řádek je zakončen */. Komentář se vztahuje na nejbližší logickou část struktury kódu (proměnná, funkce, třída, soubor).

Každý takovýto komentář lze rozdělit na 3 části:

- Krátký popis
- Dlouhý popis
- Tagy

Krátký popis je jednoduchý, většinou jednořádkový (maximálně však 3 řádky). Velmi stručně charakterizuje komentovanou část. Je zakončen tečkou nebo prázdným řádkem.

Dlouhý popis je určen k podrobnějším informacím o chování popisované části kódu. Jeho délka není omezena a může obsahovat povolené formátovací značky HTML (například pro tučné písmo, kurzívu a podobně). Pomocí tagů a jej lze rozdělit na více odstavců.

Tagy informují parser o způsobu prezentace informací a jejich zobrazení v dokumentaci. Všechny jsou jednoslovné a začínají symbolem „@" na novém řádku. Žádny tag není povinný. Umožňují do komentářů vložit informace o autorovi, verzi, parametrech funkce a její návratové hodnotě, odkazy a podobně. Některé tagy lze pomocí syntaxe {@tag text} vložit přímo do textu, bez nutnosti odsazení na nový řádek.

Příklad PHPDoc komentáře:

```
/**
* Stručný popis funkce get
*
* Podrobný popis funkce get na více řádku,
* například na tři. Tento komentář obsahuje „inline" odkaz
* na jinou {@link Uzivatel::odhlas()} část dokumentace.
*
* @param string jméno objektu, který chci vrátit
* @return object vrácený objekt
* @author Jindřich Kuchař <jindra.kuchar@hotmail.com>
*/
function get($name) { … }
```

5.1.1 Základní tagy PHPDoc

Následující tagy jsou používaný při vývoji UASF:

@author Autor popisovaného elementu. Do lomených závorek lze vložit autorův email.
@link Odkaz na URL v dokumentaci. Tento tag může být vložen přímo do textu.
@param Popisuje parametr a jeho datový typ
@return Návratový typ funkce nebo metody.
@see Zobrazí odkaz na dokumentaci libovolného elementu
@uses Zobrazí odkaz na dokumentaci libovolného elementu a vytvoří v něm zpětný odkaz
@version Verze popisovaného elementu

5.2 phpDocumetor

phpDocumetor je nástroj s otevřeným kódem, určený pro automatické generování dokumentace z komentářů PHPDoc. Lze jej využít z příkazového řádku nebo pomocí webového rozhraní. Na vstupu očekává adresář se zdrojovými soubory, na výstupu dává dokumentaci ve formátu HTML, XML, PDF nebo CHM.

5.3 Javadoc a komentáře „doc"

Javadoc je nástroj pro generování dokumentace API ve formátu HTML ze zdrojového kódu napsaného v jazyce Java. Pro svou práci využívá „doc" komentáře, které jsou (až na pár výjimek) stejné jako u PHPDoc (který z „doc" komentářů vychází). Za vývojem tohoto nástroje stojí firma Sun Microsystem, nyní vlastněna firmu Oracle. Některá vývojová prostředí (například NetBeans) automaticky využívají „doc" pro usnadnění práce programátorům a pro generování HTML dokumentace.

5.4 Generování dokumentace pro kód v jazyce C#

Sada Visual Studio 2008 podporuje pro vytváření dokumentace formát založený na XML. Dokumentaci lze vygenerovat pomocí kompilátoru s použitím parametru `/doc`. Výsledkem je soubor, který lze pomocí XSL převést do libovolného formátu, jako je například HTML. Mnohé vývojové prostředí také využívají tento soubor pro zobrazení kontextové nápovědy. Vzhledem k tomu, že se pro dokumentaci využívá formát XML, mohou programátoři vytvářet vlastní sady tagů, existuje však již předpřipravená skupina, jejíž tagy mají speciální význam a jejich použití se doporučuje. A to z toho důvodu, že kompilátor u těchto tagů provádí kontrolu, a v případě chyby zobrazí varování. Dokumentační komentáře jsou vkládány přímo před dokumentovaný blog a na každém řádku začínají symboly `///`.

Příklad dokumentačního komentáře:

```
/// <summary>Krátký popis metody </summary>
/// <remarks>Podrobnější popis metody, třeba
/// na dva řádky. </remarks>
/// <param name="vstup"> Řetězec určený ke zpracování</param>
/// <returns>Vráti počet písmen v řetězci</returns>
public int mojeMetoda(string vstup) { … }
```

5.4.1 Základní dokumentační tagy XML

Následující tagy jsou použity při vývoji USAF:

`<exception cref=""></exception>` Popisuje výjimky, které muže dokumentovaný blok vyvolat. Parametr *tref* je povinný a kompilátor kontroluje, zda daná výjimka v kontextu aplikace existuje.
`<param name=""></param>` Popis parametru. Kompilátor kontroluje, zda parametr existuje.
`<remarks></remarks>` Podrobný popis komentovaného bloku.
`<returns></returns>` Popis návratové hodnoty metody.
`<summary></summary>` Stručný popis komentovaného bloku.
`<value></value>` Popis vlastnosti.

6 Jádro vyvíjené webové aplikace

Úkolem jádra vyvíjené webové aplikace je zajistit základní funkčnost a připravit nejčastěji používané třídy a funkce, které lze poté využívat pro další funkčnost. Mezi hlavní úkoly patří:

- Konfigurace aplikace
- Zachycení a logování chyb
- Automatické načítání skriptu obsahující kód tříd
- Rozlišení vývojového a uživatelského režimu
- Výchozí funkce pro zachytávání výjimek (pokud nejsou zachyceny pomocí `try – catch`)
- Připojení k databázi, práce s databází
- Práce s uživateli (registrace, práva, přihlášení)
- Funkce a třídy pro zabezpečení

6.1 Adresářová struktura

Z praktických i bezpečnostních důvodu je třeba správně navrhnout adresářovou strukturu, určit které adresáře obsahují skripty pro zpracování dat z formuláře, které adresáře jsou určené pro třídy a podobně. Výpis základní adresářové struktury vypadá následovně:

```
/
/classes
/classes/directInclude
/classes/{typ třídy}
/functions
/graphics
/css
/javaScript
/logs
/required
/scripts
/scripts/{typ skriptu}
```

Adresář `/classes/directInclude` obsahuje třídy, které jsou potřeba pro každou stránku a má tedy smysl načítat je přímo.

Adresáře `/classes/{typ třídy}` obsahuje adresáře, které mají název založený na posledních třech písmenkách názvu tříd, které obsahují. Díky tomu lze tyto třídy automaticky načítat až v případě potřeby (vytvoření instance, volání statické metody).

Adresář `/functions` obsahuje soubory s funkcemi (jeden soubor odpovídá jedné funkci).

Adresáře `/graphics`, `/css` a `/javaScript` slouží pro potřeby vzhledu aplikace.

Adresář `/logs` obsahuje všechny soubory logu generované webovou aplikací.

Adresář `/required` obsahuje skripty, které zajišťují důležité součástí generování stránky. Tyto skripty se podle potřeby vkládají na začátek nebo na konec každé stránky. Zajišťují volání funkcí, načtení potřebných souborů, nastavení a podobně.

Adresáře `/scripts/{typ skriptu}` obsahují skripty, které přijímají data z formulářů a provedou potřebnou logiku. Adresáře jsou pojmenovány podle typu skriptů, které obsahují. Skripty zpracovávající data z formulářů týkajících se uživatelů jsou tak například uloženy v adresáři `/scripts/user`.

6.2 Konfigurace aplikace

V tomto případě se jako konfigurace aplikace rozumí především cesty k adresářům, důležitým souborům a přihlašovací údaje k databázi. Výběr a nastavení vzhledu řeší související bakalářská práce studenta Michala Hrabálka.

V praxi řeší toto nastavení statická třída Config a její statické proměnné. Pro nastavení všech statických proměnných je třeba zavolat metodu `Config::init()`. Tato metoda je volána automaticky při načtení souboru obsahující třídu.

6.3 Zachycení a logování chyb

Ve výchozím nastavení se veškeré chyby zobrazují v prohlížeči uživateli. To je výhodné při vývoji, ne však v ostrém nasazeni, ve kterém mohou chyby odhalit slabiny systému a zjednodušit tak útočníkovi napadení webu. Proto jsou ve vyvíjené webové aplikaci rozlišeny dva režimy:

- Vývojový – veškeré chyby jsou zobrazovány v internetovém prohlížeči
- Uživatelský – žádné chyby nejsou zobrazovány, ale jsou uloženy do log souboru. V případě, že chyba zabraňuje dokončení zobrazení stránky, je uživatel přesměrován na stránku oznamující, že došlo k chybě. Tato chyba není uživateli nijak upřesněna.

Pro logování chyb existuje v systému statická třída Log, mající jedinou metodu `Log::log()`. Tato třída vloží do určeného souboru nový řádek obsahující řetězec předaný jako parametr.

Pro zachycení chyb obsahuje webová aplikace funkci `errorHandler()`, která nahrazuje výchozí funkci pro zachycení chyb v PHP.

Pro zachycení neošetřených výjimek slouží funkce `exceptionHandler()`, která přesměrovává ošetření chyb na funkci `errorHandler()`.

Vývojový režim je určen hodnotou *TRUE* proměnné `Config::$DEVELOPERMODE`.

6.4 Automatické načítání tříd

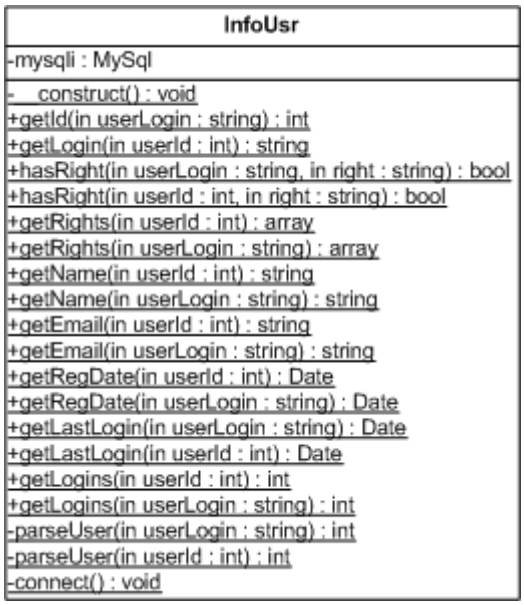
Pro automatické načítání souborů obsahujících kód jednotlivých tříd obsahuje PHP funkci `__autoload()`, která jako parametr očekává název třídy. Tato funkce je volána automaticky, a to až při prvním požadavku na konkrétní třídu. Díky tomu klesá počet zahrnovaných souborů, což zvyšuje

rychlost a omezuje množství chyb vzniklých chybným zahrnutím. Tato funkce je volaná pouze v případě, že je v aplikaci definována (může být definována pouze jednou).

6.5 Práce s uživateli

Pro přímou práci s uživateli slouží dvě třídy:

Statická třída InfoUsr – umožňuje získat informace o uživateli (jako například id, login, jméno, email a podobně) pomocí statických metod, které jako parametr ve většině případu očekávají login nebo id uživatele:



Třída InfoUsr 6-1

Třída UserUsr – slouží pro vytváření objektu konkrétního uživatele. Obsahuje také statické metody pro registraci uživatele, reset hesla a podobně.

6.5.1 Třída UserUsr

Tato třída slouží pro vytvoření objektu uživatele a práci s tímto objektem. Zatímco statická třída InfoUsr se při žádosti o kterýkoliv parametr uživatele dotazuje na databázi, tato třída pro zvýšení efektivity a snížení nároků na zdroje tyto dotazy optimalizuje. Při přihlášení uživatele jsou tak z databáze načteny pouze login, id, jméno a práva uživatele. Tyto hodnoty třída drží v rámci privátních proměnných. Ostatní vlastnosti uživatele jsou načteny z databáze teprve v případě potřeby. Po načtení jsou uloženy v privátních proměnných a není tedy třeba znovu volat dotazy na databázi.

Přihlášení uživatele (metoda `login()`) se skládá ze čtyř kroků:

1. Ověření, zda uživatel již není přihlášen
2. Vyhledání uživatele se zadaným login a heslem
3. Ověření, zda má uživatel právo pro přihlášení
4. Nastavení login, id, jména a práv

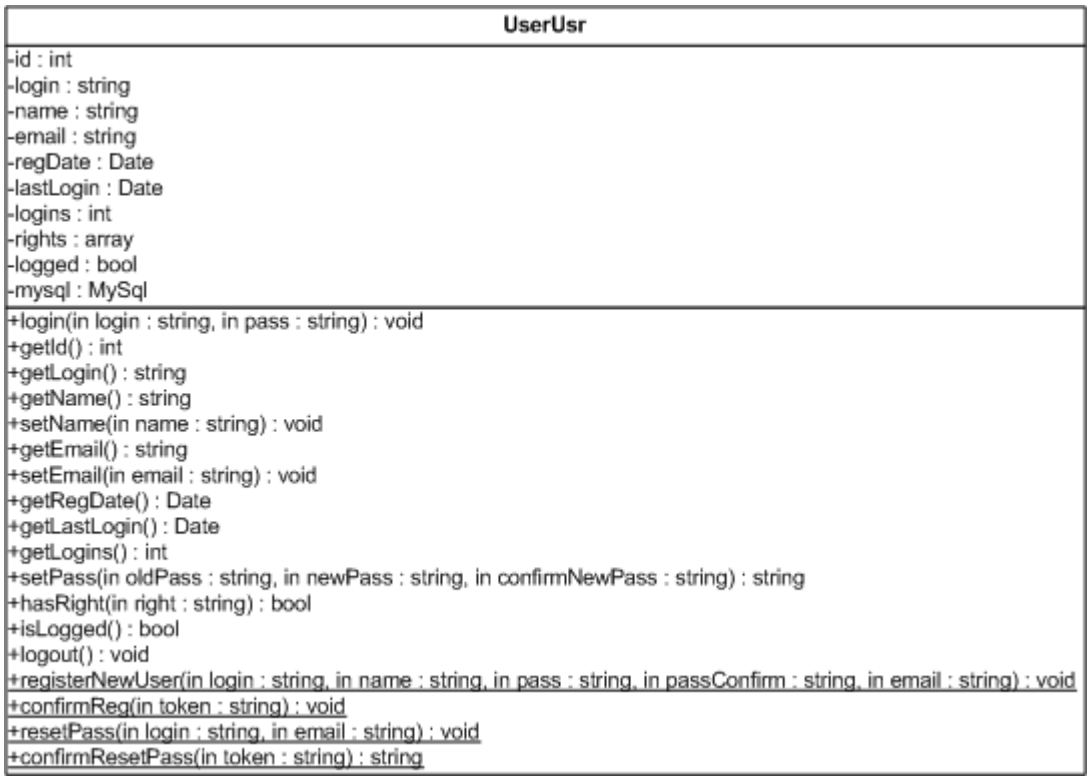
Pokud některý z bodu 1 – 3 znemožní přihlášení uživatele, je vygenerovaná výjimka s popisem chyby.

Při registraci nového uživatele je volaná statická metoda `registerNewUser()`. Postup registrace nového uživatele je následující:

1. Ověření, zda jsou vyplněny všechny potřebné položky
2. Ověření, zda souhlasí kontrola hesla
3. Ověření zadaní platného emailu (ověření tvaru emailové adresy)
4. Vložení nového záznamu
   4.1. Pokud jíž daný login existuje, dojde k vyjímce díky jedinečnému klíči na sloupci login
5. Uložení tokenu pro ověření emailu
6. Zaslání ověřovacího tokenu na email

Pro dokončení registrace musí uživatel kliknout na odkaz v zaslaném emailu.

Pokud uživatel zapomene heslo, zadá login a email zadaný při registraci. Na tento email je zasláno nové heslo a odkaz, na který je třeba kliknout pro nastavení nového hesla. Pokud uživatel na tento odkaz neklikne do 12 hodin od žádosti, nové heslo nebude aktivováno a uživatel může nadále využívat své staré heslo.



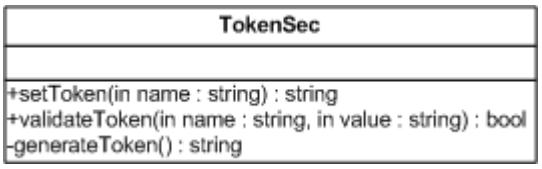
Třída UserUsr 6-1

6.6 Třídy a funkce pro zvýšení bezpečnosti

V prostředí internetu existuje velké množství útoků, proti kterým je třeba zabezpečit webovou aplikaci. Pro zvýšení bezpečnosti je důležité omezit hlášení chyb pro běžného návštěvníka

(viz Zachycení a logování chyb), zabránit SQL Injection (viz část Bezpečnost internetových aplikací) a podobně.

Jedním z útoků je také CSRF. Nejúčinnější ochranou proti tomuto útoku jsou tzv. hlídací lístky, neboli tokeny. Ve vyvíjené webové aplikaci je pro ulehčení práce s tokeny samotná třída, která umožňuje jednoduše nastavit i zkontrolovat platnost tokenu pro konkrétní operaci:



Třída TokenSec 6-2

Funkce `setToken()` slouží pro nastavení tokenu pro konkrétní operaci (podle jména). Funkce `validateToken()` slouží pro kontrolu tokenu. Po kontrole je token smazán (a to i v případě, že ověření bylo negativní).

7 Import akcí z komunitního webu www.signaly.cz

Server www.signaly.cz je křesťanský komunitní web, který umožňuje například vytvářet blogy, fotoalba, diskutovat a vkládat pozvánky na akce. A právě tyto pozvánky umožňuje exportovat pro použití jinde. Export probíhá ve formátu iCalendar.

7.1 Formát iCalendar

Formát iCalendar je standardní formát (RFC5545) pro výměnu kalendářových dat. Je podporován širokou skupinou aplikací, jako jsou například programy Mozilla Sunbird, Windows Calendar a podobně.

Distribuce dat kalendáře probíhá jako strukturovaný textový soubor s příponou .ics. Každý takovýto datový soubor začíná značkou `BEGIN:VCALENDAR` a končí značkou `END:VCALENDAR` na samostatném řádku. Mezi těmito značkami se nachází informace o vlastnostech kalendáře a jedna nebo více kalendářových komponent, jakou jsou například `VEVENT`, `VTODO` a podobně.

Informace o kalendáři musí vždy obsahovat verzi kalendáře a identifikátor tvůrce kalendáře. Mezi další vhodné parametry patří typ kalendáře (například gregoriánský, juliánský) a časové pásmo.

Přestože kalendář může obsahovat několik typů události, volny a obsazený čas a podobně, pro potřeby exportu akcí ze serveru www.signaly.cz je využito pouze komponenty `VEVENT`. Ta umožňuje definovat začátek a konec události, opakování (denně, týdně, měsíčně, ročně), stručný popis, místo konání, podrobný popis a mnohé další. Příklad kalendáře s jednou událostí vypadá takto:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Signály.cz//NONSGML Signaly.cz//EN
CALSCALE:GREGORIAN
BEGIN:VEVENT
DTSTART:20100811T170000Z
DTEND:20100811T200000Z
SUMMARY:Setkání všech signálníků
END:VEVENT
END:VCALENDAR
```

Kalendář obsahuje událost „Setkání všech signálníků“, která začíná 11. 8. 2010 v 17:00 hodin a končí tentýž den ve 20:00 hodin.

7.2 Způsob exportu akcí z www.signaly.cz

Server www.signaly.cz umožňuje dva způsoby omezení exportovaných akcí:

- Podle specifických štítků
- Podle přihlášeného uživatele

Pro vyvíjenou internetovou aplikaci je důležitější první způsob. Kalendář obsahující akce v diecézi ostravsko-opavské lze získat takto:

```
http://apps.signaly.cz/ical/events/?tag=dieceze-ostrava-opava
```

7.3 Akce ve vyvíjené internetové aplikaci

Vyvíjená internetová aplikace umožňuje dva způsoby vkládání pozvánek na nejrůznější akce:

- Přes formulář
- Importem z jednoho nebo více souborů .ICS (formát iCalendar)

V případě vkládání přes formulář jsou pozvánky uloženy v odpovídající tabulce v databázi. Z té jsou při zobrazení vybrány a zobrazeny podle data začátku. Tyto pozvánky je možné dále editovat a mazat (pokud má uživatel přidělená příslušná práva).

Při importu pozvánek ze serveru www.signaly.cz ve formátu .ICS jsou pozvánky začleněny mezi pozvánky načtené z databáze. Nelze s nimi však nadále pracovat (ani editovat, ani mazat).

8 Aplikace pro správu intencí

Součástí této bakalářské práce je také program určený pro systém Windows, jehož cílem je usnadnit správu intencí na jednotlivé bohoslužby a tisk ohlášek, které se ve většině farností čtou na konci nedělní mše. V současné době jsou pro tyto účely využívané nejrůznější postupy, jako například ruční psaní nebo sešity programu Microsoft Office Excel, které pomocí vzorců v jednotlivých buňkách práci usnadňují.

8.1 Požadavky na program

Nejdůležitějším požadavkem programu je jeho uživatelská přívětivost. Předpokládá se, že s programem budou pracovat i uživatele s pouze základní znalosti práce na počítači. Veškeré nabídky a texty tedy musí být v českém jazyce, program nesmí obsahovat žádné složité nastavení a základní funkce jako ukládání, otevírání a tisk musí být snadno dostupné tam, kde je uživatelé zvyklí na ostatní programy určené pro systém Windows očekávají (nabídka *Soubor*). Pro tyto funkce by také měly byt dostupné zažité klávesové zkratky.

Vzhledem k tomu, že ve farnostech jsou časy bohoslužeb odlišné, musí program snadno umožnit tyto časy nastavit při prvním spuštění. Důležitá je také podpora budoucí změny času bohoslužby (na příklad pro střídání letního a zimního období). V jeden den může být ve farnosti více bohoslužeb (to platí zejména pro neděli). Z důvodu přehlednosti při tisku bude počet bohoslužeb omezen na tři v neděli a dvě ve všední den.

V katolické církvi existuje velké množství svátku, které mají pohyblivé datum (například Velikonoce). Existují také svátky, na které bývá vhodné věřící upozornit, jako jsou svátky významných světců (na příklad sv. Václav nebo sv. Vojtěch). Program by měl tyto svátky automaticky doplňovat. Soubor obsahující tyto důležitá data bude program v případě potřeby stahovat z internetu.

Nedílnou součástí nedělních bohoslužeb bývají ohlášky. Jedná se o pozvánky na různé akce, důležité informace ze života farnosti a podobně. Protože některé ohlášky bývají každý týden stejné (na příklad časy výuky náboženství), program by měl umět tyto ohlášky uložit a umožnit jejich jednoduché vkládání a následný tisk. Zatímco intence se vkládají pro konkrétní den, ohlášky se vážou k celému týdnu.

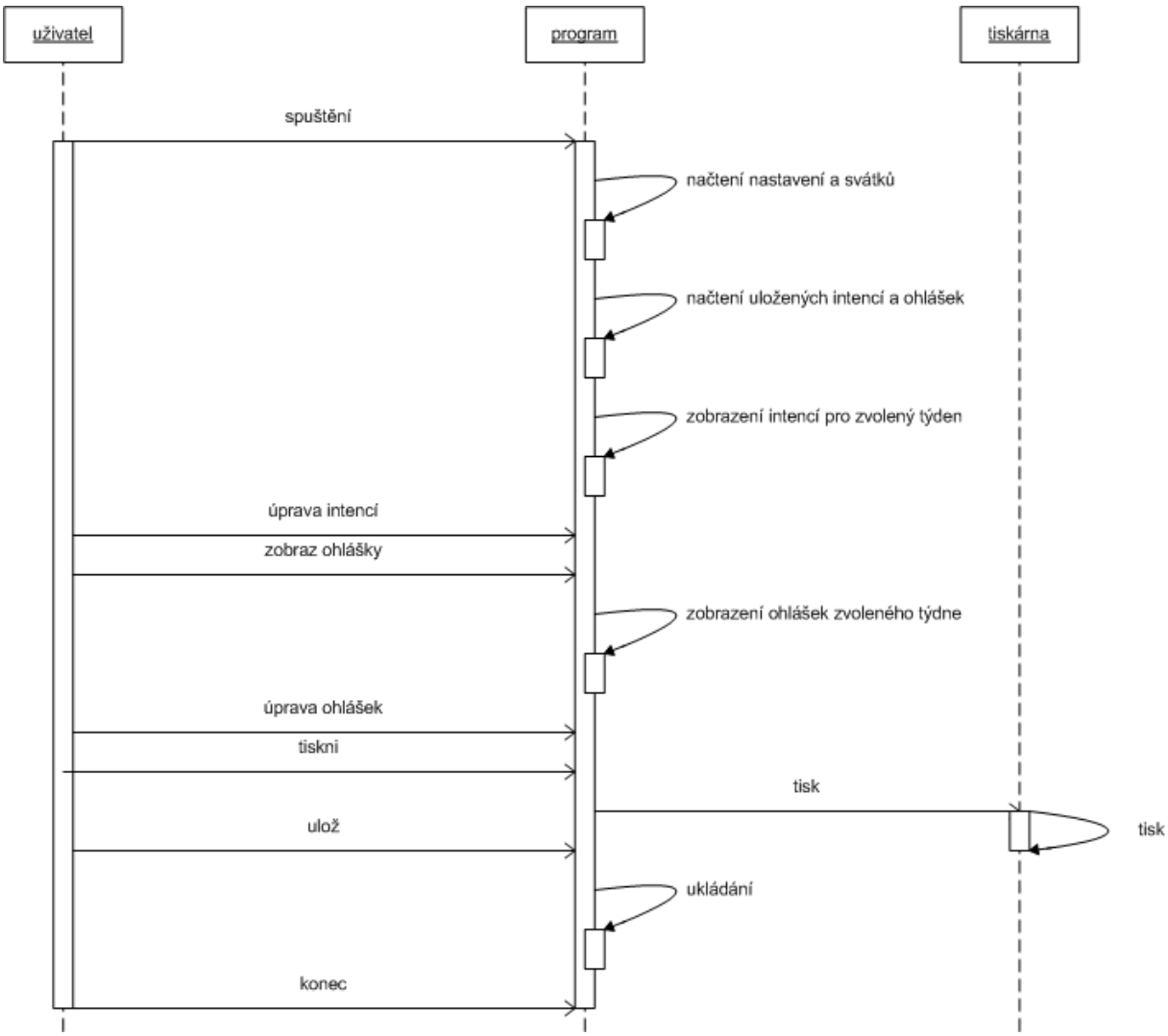
Důležitou součásti programu je také tisk intencí, času bohoslužeb, svátků a ohlášek. Ten má mít podobu tabulky o osmi řádcích (od aktuální neděle do příští neděle). První sloupec tabulky obsahuje název dne a datum, druhý sloupec obsahuje důležitý svátek na tento den, třetí sloupec obsahuje časy bohoslužeb a čtvrtý intence pro jednotlivé bohoslužby. Pod tabulkou je prostor určený pro ohlášky na aktuální týden.

Aby bylo možné zadat intence na několik týdnu dopředu, umožní program výběr týdne, se kterým se aktuálně pracuje (a to i zpětně).

8.2 Návrh implementace

8.2.1 Příklad typického užití aplikace

Uživateli se po spuštění aplikace zobrazí dialogové okno, které umožní zadat intence na další týden. Toto dialogové okno také obsahuje důležité svátky. Uživatel upraví intence a zobrazí si dialogové okno, které umožňuje zadat ohlášky na zvolený týden. Napíše nové ohlášky, případně vloží uložené. Po dokončení vše uloží a vytiskne. Poté program ukončí.



Typické použití aplikace 8-1

8.2.2 Nastavení aplikace

Pod pojmem nastavení aplikace se rozumí zadání časů bohoslužeb, jejich uložení a načtení při spuštění aplikace. Dále úpravy uložených ohlášek a načtení svátků. Při prvním spuštění aplikace po instalaci program uživatele k zadání času vyzve. Tyto časy budou uloženy v souboru XML, který budou moci

uživatelé editovat pomocí grafického rozhraní aplikace. Samotný XML soubor bude velmi jednoduchý:

```
<services>
 <day name="mon">7:00, 18:00</day>
<day name="tue">7:00</day>
…
</services>
```

V případě více bohoslužeb během jednoho dne, jsou jednotlivé časy odděleny čárkou. Po načtení těchto časů je třeba zobrazit formuláře pro zadaní intencí tak, aby odpovídaly času bohoslužeb. Toto nastavení je také třeba provést znovu po aktualizaci časů bohoslužeb. Jednotlivé procesy tedy budou vypadat takto:

1. Načtení času bohoslužeb
   1.1. Načtení souboru XML
   1.2. Získání časů pro jednotlivé dny (pondělí – neděle)
   1.3. Vykreslení formuláře s intencemi
2. Uložení (aktualizace) času bohoslužeb
   2.1. Získání časů z formuláře
   2.2. Zapsání časů do souboru XML
   2.3. Překreslení formuláře s intencemi tak, aby odpovídal novým časům

Podobný systém bude platit také pro uložené ohlášky (s tím rozdílem, že při jejich přidání není třeba znovu vykreslovat formulář pro zadání intencí). Pro uložení ohlášek bude opět zvolen soubor XML s takovouto strukturou:

```
<notes>
<note>Náboženství pro dospělé - úterý 19:30</note>
</notes>
```

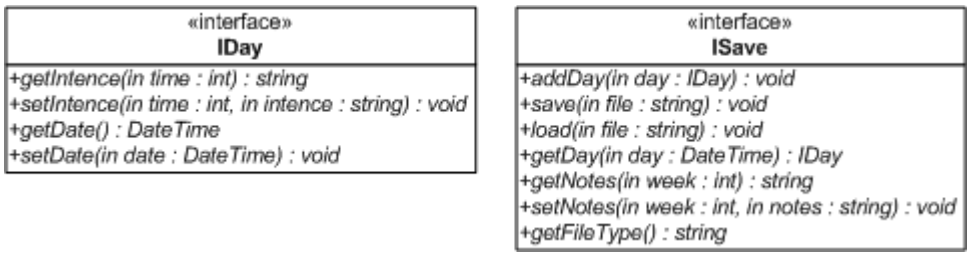
Také svátky budou uloženy v souboru XML. Tento soubor však uživatelé nebudou nijak upravovat, pouze si jednou za rok stáhnou aktualizaci obsahující svátky na další rok. Na rozdíl od předchozích XML souborů, které stačilo při spuštění aplikace jednou projít od začátku do konce (a načíst uložené údaje), v tomto souboru bude třeba vyhledávat jednotlivé svátky vždy pro konkrétní den (například pomocí XPath). XML soubor bude vypadat takto:

```
<calendar version="1">
 <day date="2010-01-01"> Památka sv. Fabiána a Šebestiána</day>
</calendar>
```

8.2.3 Úprava intencí a ohlášek

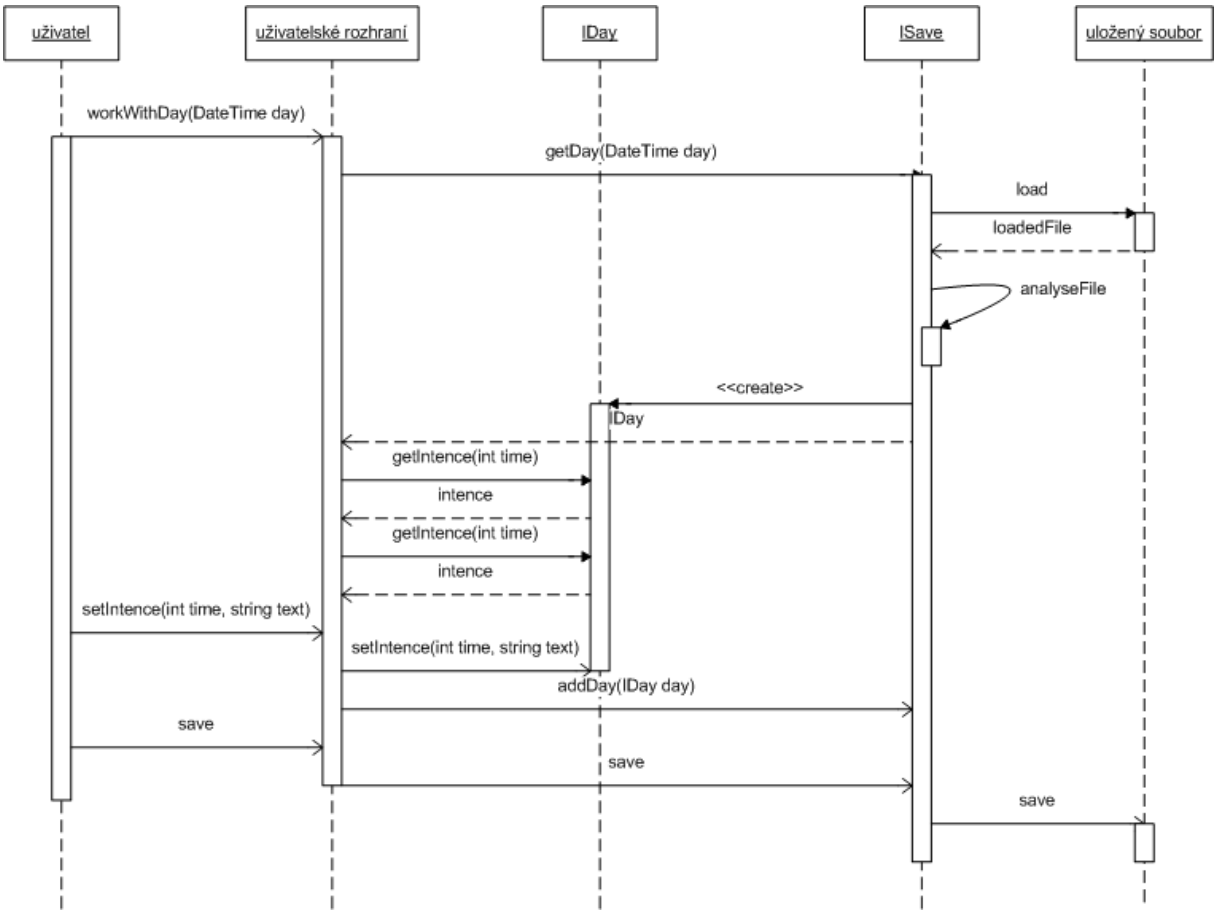
Po spuštění programu před sebou uživatel uvidí intence na nadcházející týden (od neděle do následující neděle). Zobrazený týden bude možné měnit. Pro zvolený týden se ze souboru intencí načtou intence na jednotlivé bohoslužby. Uživatel tyto intence upraví podle potřeby. Stejně tak může upravit ohlášky pro zvolený týden. Vše se pak uloží do jednoho souboru. Tento soubor bude časem obsahovat intence a ohlášky za více roků, je tedy třeba zvolit efektivní metodu ukládání tak, aby

nebylo třeba vždy generovat celý soubor. To bude zajištěno pomocí dvou tříd, jedné implementující rozhraní IDay (reprezentující aktualizovaný den) a ISave (třída, která bude spravovat objekty IDay a realizovat práci s uloženým souborem):



Rozhraní IDay a ISave 8-2

Uživatelské rozhraní zobrazí konkrétní den, pro který si od objektu typu ISave vyžádá objekt typu IDay obsahující intence. Objekt typu ISave načte soubor z disku, vyhledá v něm konkrétní den a vrátí jej. Po provedení úprav uživatelem je volaná metoda `ISave.addDay()`, která přidá nebo aktualizuje (pokud již existuje) konkrétní den v souboru. Nakonec je soubor uložen.



Práce s intencemi 8-3

Podobným způsobem bude probíhat práce s ohláškami. K tomu slouží funkce `ISave.getNotes()` a `ISave.setNotes()`. Intence budou v souboru uloženy podle data, ohlášky podle čísla týdne.

8.3 Implementace

Pro implementaci byl zvolen programovací jazyk C# spolu s technologií .NET Framework firmy Microsoft.

8.3.1 Technologie .NET Framework

Technologie .NET Framework firmy Microsoft je prostředí pro běh aplikací, určené pro systémy Microsoft Windows. Obsahuje sadu knihoven řešící velké množství častých programátorských problému a činností, jako jsou prvky uživatelského rozhraní, práce se soubory, sítí, databázemi a podobně. Součástí .NET Frameworku je také Common Language Runtime, který poskytuje funkčnost velmi podobnou virtuálnímu stroji. Do jazyka CLR jsou také kompilovány veškeré zdrojové kódy. Proto je pro tvorbu aplikací možné použít více programovacích jazyků, včetně kombinace několika různých pro jeden projekt.

8.3.2 Programovací jazyk C#

C# je objektově orientovaný, typově bezpečný programovací jazyk s podporou automatické správy paměti, podporou řízení toku programu pomoci výjimek vyvinutý firmou Microsoft pro potřeby vývoje aplikací pro .NET Framework. Svou syntaxí vychází z jazyků C++ a Java.

8.3.3 Třídy pro práci s XML v jazyce C#

Jazyk C# nabízí pro práci se soubory XML tři základní třídy: XmlReader (pro jednoprůchodové dopředné přečtení XML dokumentu), XmlWriter (jednosměrné generování XML dokumentu) a XmlDocument (reprezentuje XML dokument v paměti, implementuje DOM podle W3C).

Pro načtení souboru XML obsahujícího časy bohoslužeb postačí vzhledem k jeho jednoduchosti třída XmlTextReader. Ta načte XML ze souboru (v případě že soubor nenajde, vyzve uživatele k zadání časů bohoslužeb), pomocí cyklu jej projde a načte časy.

Při ukládání časů je využita třída XmlWriter, která vygeneruje XML dokument a uloží jej do zvoleného umístění.

Stejně jako časy bohoslužeb funguje i ukládání a načítání uložených ohlášek.

Jiná situace je u souboru XML obsahujícího svátky. Zde již nepostačí jednosměrný průchod celým souborem. Proto je využita třída XmlDocument, která umožňuje vyhledat konkrétní den pomocí výrazu XPath.

8.3.4 Načítání a ukládání intencí

Pro realizaci načítání a ukládání intencí byl zvolen model zásuvných modulů. Tím je do budoucna umožněno zvolení rozdílného způsobu uložení, který bude odrážet budoucí požadavky uživatelů. Z důvodu bezpečnosti poběží zásuvné modely v rozdílné aplikační doméně. Jednotlivé moduly jsou

reprezentovány souborem .DLL umístěným v adresáři *plugins/*. Při spuštění aplikace program nejdříve projde tento adresář a u každého souboru .DLL zjistí, zda implementuje rozhraní ISave. Teprve potom vytvoří příslušný objekt, který realizuje samotné načítání a ukládáni. Podle přípony souboru intencí je volán správný zásuvný modul.

V instalaci programu je obsažen pouze jeden zásuvný modul, který realizuje ukládání intencí do souboru XML. Ten udržuje v paměti jako objekt třídy XmlDocument. Při volání funkce `ISave.addDay()` se nejdříve pokusí vyhledat uzel s určeným datem a pokud existuje, aktualizuje jej. Pokud neexistuje, vytvoří pro zadané datum uzel nový.

9 Závěr

Tato práce byla zaměřena na možnosti vývoje webové aplikace a její vývoj. Po představení dostupných technologií jsem podrobněji rozebral možnosti využité při tvorbě webové aplikace, jako je PHP, MySQL, a technologie .NET firmy Microsoft.

Část práce jsem věnoval spoluprácí PHP a databáze MySQL, naznačil možnosti jednotlivých řešení, jejich výhody a nevýhody. Část věnovaná databázím také obsahuje testy rychlosti jednotlivých možností.

Důležitou součástí každé webové aplikace je její zabezpečení. Z toho důvodu jsem provedl seznámení s nejčastějšími typy útoku, uvedl jejich praktické příklady a popsal obranu.

Z praktické části jsem načrtl základní součásti jádra vyvíjené webové aplikace, jako je práce s uživateli nebo třídy pro zvýšení bezpečnosti. Dále byl probrán import pozvánek na akce ze serveru www.signaly.cz (ve formátu iCalendar).

V rámci vývoje aplikace pro systém Windows jsem provedl seznámení s .NET Frameworkem firmy Microsoft, provedl analýzu typického použití, navrhl konkrétní řešení pomocí tříd .NET Frameworku a celou aplikaci prakticky realizoval.

Pro usnadnění budoucího vývoje a případné zapojení dalších vývojářů pojednává tato práce o možnostech programátorské dokumentace.

Nasazení webové aplikace bude uskutečněno na objednávku konkrétních zájemců. Aplikace umožňuje provádět nejčastěji požadované úkony, v případě zájmu by však neměl být problém doplnit novou funkčnost.

10 Bibliografie

.NET Framework: Overview. *.NET Framework.* [Online] c2009. [Citace: 8. 4 2010.] http://www.microsoft.com/net/overview.aspx.

About. *python.* [Online] c1990-2010. [Citace: 24. 2 2010.] http://www.python.org/about/.

Alshanetsky, Ilia. 2005. *PHP|Architect's guide to PHP Security.* Toronto : Marco Tabini & Associates, Inc., 2005. ISBN 0-9738621-0-6.

Fuecks, Harry. 2003. *The PHP Anthology Volume I: Foundations.* Collingwood : SitePoint Pty. Ltd., 2003. ISBN 0-9579218-5-3.

Fuecks, Harry. 2003. *The PHP Anthology Volume II: Applications.* Collingwood : SitePoint Pty. Ltd., 2003. ISBN 0-9579218-4-5.

Gilmore, Jason W. 2008. *Beginning PHP and MySQL.* New York : apress, 2008. ISBN 978-1-59059-862-7.

Gutmans, Andi, Bakken, Stig Seather a Rethans, Derick. 2007. *Mistrovství v PHP 5.* Brno : Computer Press, a.s., 2007. ISBN 978-80-251-1519-0.

Howard, Michael a LeBlanc, David. 2008. *Bezpečný kód.* Brno : Computer Press, a.s., 2008. ISBN 978-80-251-2050-7.

Informační systém - Wikipedie, otevřená encyklopedie. *Wikipedia.* [Online] 3. 31 2010. [Citace: 1. 10 2010.] http://cs.wikipedia.org/wiki/Informa%C4%8Dn%C3%AD_syst%C3%A9m.

Lacko, Luboslav. 2005. *PHP a MySQL Hotová řešení.* Brno : CP Books, a.s., 2005. ISBN 80-251-0397-8.

Mejzr, Martin. 2008. Bakalářská práce na fakultě elektrotechniky Českého vysokého učení technického v Praze. *Bezpečnostní problémy internetových aplikací.* Praha : Vedoucí práce Ing. Zdeněk Troníček Ph.D., 2008.

PHPDoc - Wikipedia, the free encyclopedia. *Wikipedia.* [Online] 22. 4 2010. [Citace: 27. 4 2010.] http://en.wikipedia.org/wiki/Phpdoc.

RFC 5545 - Internet Calendaring and Scheduling Core Object Specification (iCalendar). *IETF Tools.* [Online] 9 2009. [Citace: 26. 2 2010.] http://tools.ietf.org/html/rfc5545.

Schlossnagle, George. 2004. *Advanced PHP Programming.* Indianapolis : Sams Publishing, 2004. ISBN 0-672-32561-6.

Snyder, Chris a Southwell, Michael. 2005. *Pro PHP Security.* New York : apress, 2005. ISBN 1-59059-508-4.

Sweat, Jason E. 2005. *PHP|Architect's guide to design patterns.* Toronto : Marco Tabini & Associates, Inc., 2005. ISBN 0-9735898-2-5.

Systém řízení báze dat - Wikipedie, otevřená encyklopedie. *Wikipedia.* [Online] 1. 4 2010. [Citace: 10. 4 2010.]
http://cs.wikipedia.org/wiki/Syst%C3%A9m_%C5%99%C3%ADzen%C3%AD_b%C3%A1ze_dat.

Trnka, Filip. Bakalářská práce na pedagogické fakultě Jihočeské univerzity v Českých Budějovicích. *Redakční systém realizovaný pomocí PHP a MySQL.* České Budějovice : Vedoucí práce Ing. Michal Šerý Autor: Filip.

Vrána, Jakub. c2005-2008. PHP triky - Cross-Site Request Forgery. *PHP triky.* [Online] c2005-2008. [Citace: 15. 1 2010.] http://php.vrana.cz/cross-site-request-forgery.php.

Zandstra, Matt. 2008. *PHP Objects, Patterns, and Practice Second Edition.* New York : apress, 2008. ISBN 978-1-59059-909-9.